MoVES - A Framework for Modelling and Verifying Embedded Systems

Brekling, Aske Wiid; Hansen, Michael Reichhardt; Madsen, Jan

Published in:
2009 International Conference on Microelectronics

Link to article, DOI:
10.1109/ICM.2009.5418667

Publication date:
2009

Document Version
Publisher's PDF, also known as Version of record

MoVES - A Framework for Modelling and Verifying Embedded Systems

Aske Brekling and Michael R. Hansen and Jan Madsen
DTU Informatics
Technical University of Denmark
2800 Kgs. Lyngby - Denmark
Telephone: (+45) 4525–3351
Fax: (+45) 4588–2673
Email: {awb,mrh,jan}@imm.dtu.dk

***Abstract*— The MoVES framework is being developed to assist in the early phases of embedded systems design. A system is modelled as an application running on an execution platform. The application is modelled through the individual tasks, and the execution platform is modelled through the processing elements, including the operating systems, and their interconnections. The tasks and processing elements are characterized by their real-time properties. The framework can be used to conduct schedulability analysis and has the potential to reason about different types of resource usage such as memory usage and power consumption. A simple specification language for embedded systems and a verification backend are presented. The framework has a modular, parameterized structure supporting easy extension and adaptation of the specification language as well as of the verification backend. We show, using a number of small examples, how MoVES can be used to model and analyze embedded systems.**

I. Introduction

Modern hardware systems are moving toward execution platforms made up of multiple programmable and dedicated processing elements implemented on a single chip, known as a *multi-core execution platform*. The different parts of an embedded application are executing on these processing elements; but the activity of mapping the parts of an embedded program onto the platform elements is non-trivial. First of all, there may be various and often conflicting resource constraints. Real-time constraint; for example, should be met together with constraints on the use of memory and energy. There also is a huge variety in freedoms of choice in the mapping of an application to a platform because a) there are many ways to partition an embedded program into parts, b) there are many ways these parts can be assigned to processing elements and c) each processing element can be set up in many ways.

As embedded systems become more complex, the interaction between application and execution platform becomes more incomprehensible and problems such as memory overflow, data loss, and missed deadlines become more likely. In the development phase it is not enough to simply look at the different layers of the system independently, as a minor change at one layer can greatly influence the functionality of other layers. System-level verification of schedulability, upper limits for memory usage, and power consumption, taking all layers into account, have therefore become central fields of study in the design of embedded systems.

As many important design decisions are made early in the design phase, it is imperative to support the system designer at this level. This paper presents the MoVES verification framework, which is based on an abstract embedded system model from the simulation based ARTS framework [8], [9]. The ARTS framework captures essential properties of a set of applications executing on a multi-core execution platform. In [3] a formal model for ARTS-based multi-core systems was defined, and it was shown how this model could be expressed by timed automata. The MoVES framework relies on this formal model to analyze embedded systems. MoVES is set up to use external verification engines such as the UPPAAL [2] model checker to verify system-specific properties e.g. schedulability.

Other tools supporting schedulability analysis of multi-core systems are SymTA [5] and real-time-calculus [10]. Both systems are based on arrival- and service curves from Network Calculus [4] and analysis is performed using over-approximations. Furthermore, TIMES [1] is a timed-automata based tool for the analysis of single-processor systems.

II. A Model of Multi-Core Systems

The MoVES model [3] consists of an application to be executed on an execution platform, which is made up of processing elements and their connections. Individual tasks make up the application together with a *task graph* showing data dependencies. Finally the application is mapped to the execution platform by mapping tasks to processing elements. The main idea of a system modelled in MoVES is shown in Fig. 1. Note that arrows with white tips in the application denote data dependencies, and arrows with black tips between the application and the execution platform denote the mapping.

A. Models of application components

In MoVES, a task is periodic and preemptive and can be described using three states:

- *Released* - the task is ready, but is not currently executing,
- *Running* - the task is executing, and

978-1-4244-5816-5/09/$26.00 ©2009 IEEE

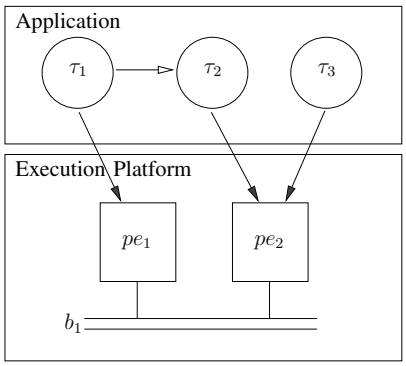

Fig. 1. Multi-core system model

- *Done* - the task has finished its job in the current period and is waiting for its release in the next period.

When a task is preempted, it moves from the state *Running* back to *Released*.

An offset for a task describes an initial number of time units that must pass before the task is released for the first time. Dependent tasks must have same periods and offsets.

B. Models of execution platform components

The basic components of an execution platform are processing elements and busses. An allocation mechanism is required when such resources are shared. For processing elements this mechanism is *scheduling* and for busses *arbitration*. The scheduling on a processing element is usually conducted by a *real-time-operating-system* (RTOS) and a specific scheduling principle, e.g. *rate monotonic* (RM), *earliest deadline first* (EDF) or *fixed priority* (FP) is chosen. An arbiter for a bus could be based on *first in - first out* (FIFO), for example.

C. Model instantiation

The best and worst-case execution times of a task depend on the processing element on which it is executing. Thus, once a mapping of an application onto an execution platform is known, the timing properties for tasks can be extracted from the characteristics of the processing elements. Furthermore, if dependent tasks are mapped to different processing elements, that dependency will result in data transfer. In such a case, the speed of the bus connecting the processing elements combined with the amount of data to be transferred provides the time for the transfer.

For a complete system specification consisting of application, execution platform and mapping, MoVES provides a timed-automata generator that can be used for verification of system properties. In [7] it was shown how this approach can be used to reason about resource usage such as memory- and power consumption. The version of MoVES presented here deals just with schedulability analysis in terms of absence of missed deadlines. MoVES also has a trace generator, which can show system runs with a missed deadline.

III. Using MoVES

MoVES provides a simple specification language for multi-processor systems. A grammar for this language is given in this section. The MoVES utility can generate a timed-automata model for a specified system as well as a query that can be used to verify the specified property. MoVES uses an external verification back-end and provides the user with the verification result, either of the form *property is satisfied* or of the form *property is NOT satisfied*. If the property is not satisfied, then MoVES can provide a violating trace. Fig. 2 provides an overview of the MoVES framework.

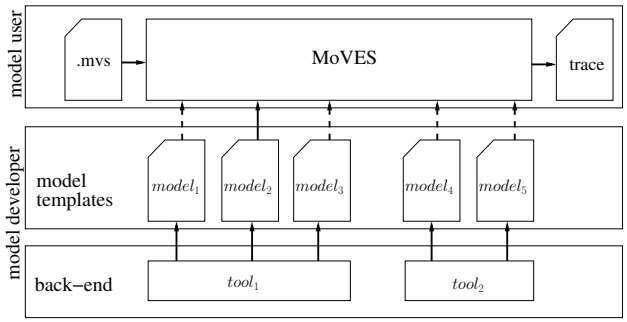

Fig. 2. Overview of the MoVES framework

A. System Specification Language

MoVES has a simple specification language for systems fitting the model for multi-core systems described in Section II. In Fig. 3, an example is given, which could be a specification of the system shown in Fig. 1. The grammar for specifications is provided is given in Fig. 4.

In the example specification in Fig. 3 we see a system consisting of an application made up of three tasks (`T1`, `T2` and `T3`), a platform with two processing elements (`P1` and `P2`), and a bus (`B1`). The tasks `T1` and `T2` both have periods 6 and offset 0, `T3` has period 4 and offset 3. There is a data dependency from `T1` to `T2` where 2 data units need to be transferred (only applicable if the tasks are mapped to different processing elements). The processing element `P1` uses RM

```
Application
 Task: T1
  Period: 6
  Offset: 0
 Task: T2
  Period: 6
  Offset: 0
 Task: T3
  Period: 4
  Offset: 3

 Dependencies
  T1 -> T2 : 2
Platform
 Proc: P1
  Sch: RM
 Proc: P2
  Sch: EDF
 Bus: B1
  Arb: FIFO
  Speed: 2

      Mapping
       T1 : P1
       T2 : P2
       T3 : P2
      Characteri
       T1 @ P1
        Bcet: 2
        Wcet: 2

       T2 @ P2
        Bcet: 1
        Wcet: 2

       T3 @ P2
        Bcet: 1
        Wcet: 2

      Property
       Schedule?
```

Fig. 3. Example specification

```
system ::= app plat map chr prop
app    ::= Application task+ dep
task   ::= taskid per off
taskid ::= Task: tid
per    ::= Period: n
off    ::= Offset: n
dep    ::= Dependencies dp*
dp     ::= tid -> tid : n
plat   ::= Platform proc+ bus
proc   ::= Proc: pid Sch: sch
sch    ::= FP | RM | EDF
bus    ::= busid arbit speed
busid  ::= Bus: bid
arbit  ::= Arb: arb
arb    ::= FIFO
speed  ::= Speed: n
map    ::= Mapping mp+
mp     ::= tid : pid
chr    ::= Characteri tonp+
tonp   ::= tid @ pid bcet wcet
bcet   ::= Bcet: n
wcet   ::= Wcet: n
prop   ::= Property p
p      ::= Schedule?

n ∈ ℕ, tid, pid and bid are strings
```

Fig. 4. MoVES specification language grammar

scheduling and P2 uses EDF scheduling. The bus B1 uses a FIFO arbiter, and operates at a speed of 2 data units for each time unit. Task T1 is mapped to P1, T2 and T3 are mapped to P2. The characteristics show that executing T1 on P1 takes 2 time units in both best and worst case, executing either T2 or T3 on P2 takes 1 in best case and 2 time units in worst case. Finally, the property we wish to verify is whether the system is schedulable.

The grammar follows the model described in Section II. A system is composed of an application, a platform, a mapping, characteristics and a property. An application consists of a number of tasks and their dependencies. A task is defined in terms of its name (or id), period and offset, and the dependencies describe a data dependency between two tasks and the amount of data to be transferred if they are mapped to different processing elements. A platform consists of a number of processing elements and a bus connecting them. A processing element is defined in terms of its name (or id) and scheduling principle, and the bus is defined by its name (or id), arbiter and the speed at which it operates. MoVES currently supports scheduling principles FP, RM and EDF and arbiter FIFO, however, other principles could easily be added. The mapping associates each task with one processing element. The characteristics give quantitative properties in terms of *best case execution time* (bcet) and *worst case execution time* (wcet) for executing the different tasks on the different processing elements specified. Finally, the property specifies which system property to verify. Currently, MoVES supports verification of schedulability only, but other properties could be introduced.

B. Framework

There are different ways in which specifications can be transformed to timed automata, and there is no known best way of doing so. Therefore, MoVES supports the use of different timed-automata representations of specifications, which are given in the form of templates as input to the MoVES tool. For the novice user, the default options can be used. Currently we provide three different templates corresponding to different ways of dealing with preemptive tasks.

Discrete running time (drt). In this model the running time is made discrete. A counter for each task is introduced to keep track of remaining execution time. The UPPAAL model checker is used as verification back-end.

Stop watches (sw). This model uses stop watches for the execution time. A developmental UPPAAL model checker supporting stop watches is used as verification engine.

No clocks (nc). In this model all clocks are removed. Instead, a list of essential time points is statically generated. These time points are the only time points at which tasks can be released. The UPPAAL model checker is used as back-end.

After saving a specification in a file `system.mvs`, MoVES verifies the specified property with the command: `moves system.mvs`. This will use the default template (the one with no clocks) and the default UPPAAL verification engine. If the verification succeeds, MoVES responds with *Property is satisfied*, if the verification fails, MoVES provides a violating trace for the property tried for verification.

If MoVES is supplied with one of the options `-drt`, `-sw` or `-nc`, the corresponding template and verification back-end will be used, e.g. `moves -sw system.mvs` will verify the system specified in `system.mvs` with the model template using stop watches, and using the developmental UPPAAL model checker supporting stop watches as back-end.

IV. EXAMPLES

We provide a few academic examples to serve as inspiration. First in Example 1, we show how the specification from Fig. 3 can be analyzed. Example 2 is an interesting single-core system, where we show how MoVES can be used for design space exploration. Finally, in Example 3, we use MoVES for analysis of multi-core systems with inter-processor dependencies.

A. Example 1

Verification of the system in Fig. 3 gives *property is satisfied*, i.e. the system is schedulable. Experimenting with different timing values, different scheduling principles, etc. could provide other results. If for example the wcet for T2 was raised to 3, the system would no longer be schedulable, and MoVES would provide a trace with a missed deadline.

B. Example 2

In this example we consider three tasks (T1, T2 and T3) on a single processor running EDF scheduling. In Fig. 5a we provide the specification.

Verification of this system gives *property is NOT satisfied*, and the following trace:

```
   | 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8
T1 | ++++++    ++++++    ++++++    ++++++    ++++++    ++++++
T2 | OOOOOO++++OOOOOO++  OOOOOOOO++OOOOOO++++OOOOOO++++OOOOOO++
T3 | OOOOOOOOOOOOOOOOO++OOOOOO++  OOOOOOOOOOOOOOOOOOOOOOOOOOOOX
```

In this trace a + indicates that the task is executing in that time unit, a 0 means that the task is released but not executing

TABLE I

CHARACTERISTICS FOR SYSTEM IN EXAMPLE 3

	T1	T2	T3	T4
P1	(2,2)	(1,1)	(4,5)	-
P2	(1,2)	-	(2,2)	(2,3)

(due to either a higher prioritized task being executed or an unresolved data dependency), an X shows that the task will miss its deadline and a blank means that the task is waiting to be released in its next period.

We see that T3 misses a deadline after 58 time units. We alter the system and lower the wcet of T3 to 3. Verification of the altered system gives *property is satisfied*. We would like to see if the processor instead can use RM scheduling. Verification of this system gives *property is NOT satisfied*, and the following trace:

```
   | 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0
T1 | ++++++    ++++++    ++++++    +
T2 | OOOOOO++++OOOOOO++  OOOOOO++++O
T3 | OOOOOOOOOOOOOOOOOO++OOOOOOOOOX
```

Examination of the trace shows that T3 misses a deadline after 30 time units. We therefore lower the wcet of T3 to 2. Verification of the system now gives *property is satisfied*.

```
Application
 Task: T1
 Period: 10
 Offset: 0

 Task: T2
 Period: 20
 Offset: 0

 Task: T3
 Period: 30
 Offset: 0

 Dependencies

Platform
 Proc: P1
 Sch: EDF

 Bus: B1
 Arb: FIFO
 Speed: 2

Mapping
 T1 : P1
 T2 : P1
 T3 : P1

Characteri
T1 @ P1
 Bcet: 6
 Wcet: 6

T2 @ P1
 Bcet: 6
 Wcet: 6

T3 @ P1
 Bcet: 3
 Wcet: 3
```

a) Example 2

```
Application
 Task: T1
  Period: 4
  Offset: 0

 Task: T2
  Period: 6
  Offset: 0

 Task: T3
  Period: 6
  Offset: 0

 Task: T4
  Period: 6
  Offset: 4

 Dependencies
  T2 -> T3 : 2

Platform
 Proc: P1
  Sch: RM

 Proc: P2
  Sch: RM

 Bus: b1
  Arb: FIFO
  Speed: 2

Mapping
 T1 : P1
 T2 : P1
 T3 : P2
 T4 : P2

Characteri
 T1 @ P1
  Bcet: 2
  Wcet: 2

 T2 @ P1
  Bcet: 1
  Wcet: 1

 T3 @ P1
  Bcet: 4
  Wcet: 5

 T1 @ P2
  Bcet: 1
  Wcet: 2

 T3 @ P2
  Bcet: 2
  Wcet: 2

 T4 @ P2
  Bcet: 2
  Wcet: 3
```

b) Example 3

Fig. 5. Example systems specifications

C. Example 3

In Example 3 we consider an application consisting of 4 tasks (T1, T2, T3 and T4), there is a data dependency from T2 to T3 with 2 data units. We wish to examine schedulability of this application on an execution platform consisting of two processing elements (P1 and P2, both using RM scheduling) connected through a bus with the speed of 2 data units pr. time unit. In Table I we give the characteristics of the system. For each pair (b,w) in the table, b is bcet and w is wcet. Hyphen symbols (-) indicate that the given task cannot be executed on that processing element.

We decide to map T1 and T2 to P1 and T3 and T4 to P2. In Fig. 5b this example system is provided as a MoVES specification.

Verification of the system gives *property is NOT satisfied*, and the resulting trace is given below (note that the task of transferring data due to the dependency from T2 to T3 is described by the task T2_T3):

```
      | 0 2 4 6 8 0
T1    | ++  ++  ++
T2    | OO+   +
T3    | OOOO++OO++
T4    |     OO++OOX
T2_T3 | OOO+  O+
```

We see that T4 misses a deadline after 10 time units. Altering the system to use EDF scheduling for P2 and doing the verification again gives *property is satisfied*; in other words, changing scheduling principle on just P2 makes the system schedulable.

V. SUMMARY

We have presented the MoVES verification framework and how it can be used to verify schedulability of multi-core embedded systems. Emphasis has been on explaining the general concepts from a users point of view. The capabilities of MoVES have been demonstrated through a number of simple examples, which are all verified within a few seconds, however, we have successfully verified parts of a smart phone application with 103 tasks executing on a 4-core execution platform, in little more than one hour. We are currently working on extending the framework to handle different types of resource usage. The MoVES framework and a set of examples can be downloaded from http://www.imm.dtu.dk/moves.

REFERENCES

[1] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson and W. Yi, *Times - A Tool for Modelling and Implementation of Embedded Systems* In proceedings of 8th International Conference, TACAS 2002, pages: 460-464, Springer-Verlag, LNCS Vol. 2280, 2002.

[2] G. Behrmann, A. David, K.G. Larsen, *A tutorial on uppaal*, Lecture Notes in Comput. Sci. 3185 (2004) pp. 200236.

[3] A. Brekling, M. R. Hansen and J. Madsen, *Models and formal verification of multiprocessor system-on-chips*, The Journal of Logic and Algebraic Programming, 77(1):1-19, Elsevier, 2008.

[4] R. L. Cruz, *A calculus for network delay. I. Network elements in isolation*, IEEE Transactions on In Information Theory, IEEE Transactions on, Vol. 37, No. 1. (1991), pp. 114-131.

[5] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, R. Ernst, *System level performance analysis - the SymTA/S approach*, IEE Proceedings Computers and Digital Techniques, 2005.

[6] J. Madsen, M. R. Hansen, A. W. Brekling, *A Modelling and Analysis Framework for Embedded Systems*, in: Model-Based Design of Heterogeneous Embedded Systems, P. Mosterman, G. Nicolescu (editors), CRC Press, 2009

[7] J. Madsen, M. R. Hansen, K. S. Knudsen, J. E. Nielsen, A. W. Brekling, *System-level verification of multi-core embedded systems using timed automata*, part of: 17th World Congress International Federation of Automatic Control (IFAC), pages: 9302-9307, 2008.

[8] J. Madsen, K. Virk, M.J. Gonzalez, *A SystemC-Based Abstract Real-Time Operating System Model for Multiprocessor System-on-Chip*, in Multiprocessor System-on-Chip, Morgan Kaufmann 2004, pp. 283-312

[9] S. Mahadevan, K. Virk, J. Madsen, *ARTS: a SystemC-based framework for multiprocessor systems-on-chip modelling*, Des. Automat. Embedded Syst. 11(4) (2007) 285-311

[10] L. Thiele, S. Chakraborty and M. Naedele, *Real-time calculus for scheduling hard real-time systems*, International Symposium on Circuits and Systems ISCAS 2000, 4 (2000) pp. 101-104.